

**INFORMATION PROCESSING SYSTEM AND STORAGE MEDIUM**

Patent Number: JP2002073345  
Publication date: 2002-03-12  
Inventor(s): ETO TAKESHI  
Applicant(s): FUJITSU LTD  
Requested Patent: JP2002073345  
Application Number: JP20000255317 20000825  
Priority Number(s):  
IPC Classification: G06F9/45  
EC Classification:  
Equivalents:

---

**Abstract**

---

**PROBLEM TO BE SOLVED:** To execute a program for a virtual machine faster.

**SOLUTION:** The program loader 16 reads out a byte code from the cluster file 12, decides whether a load location of this byte code and an argument setup byte code is the Java virtual machine 13 or the JIT compiler 15 based on content of a determination information storage unit 17 if the byte code is a subroutine call and iterates a loading process for the byte code to the decided load location. If the byte code is not the subroutine call, the loading location is decided to Java virtual machine 13. The content of determination information storage unit 17 is decided by the static analyzer 18 to analyze the content of the cluster file 16 or a person before the cluster file 12 is executed.

---

Data supplied from the esp@cenet database - I2

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開2002-73345

(P2002-73345A)

(43) 公開日 平成14年3月12日 (2002.3.12)

(51) Int.Cl.<sup>7</sup>

識別記号

F I

テーマコード (参考)

G 0 6 F 9/45

G 0 6 F 9/44

3 2 0 C 5 B 0 8 1

審査請求 未請求 請求項の数 5 O L (全 7 頁)

(21) 出願番号 特願2000-255317 (P2000-255317)

(22) 出願日 平成12年8月25日 (2000.8.25)

(71) 出願人 000005223

富士通株式会社

神奈川県川崎市中原区上小田中4丁目1番  
1号

(72) 発明者 衛藤 健

神奈川県川崎市中原区上小田中4丁目1番  
1号 富士通株式会社内

(74) 代理人 100092587

弁理士 松本 眞吉

Fターム (参考) 5B081 AA09 DD01

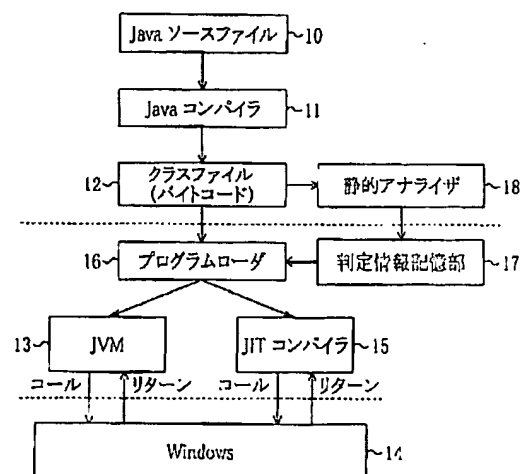
(54) 【発明の名称】 情報処理装置及び記録媒体

(57) 【要約】

【課題】 仮想マシンのプログラムをより高速に実行する。

【解決手段】 プログラムローダ16は、クラスファイル12からバイトコードを読み出し、バイトコードがサブルーチンコールであれば、判定情報記憶部17の内容に基づいてこのバイトコード及び引数設定バイトコードのロード先を、Java仮想マシン13にするかJITコンパイラ15にするかを決定し、決定されたロード先にバイトコードをロードするという処理を繰り返し実行する。バイトコードがサブルーチンコールでなければロード先をJava仮想マシン13にする。判定情報記憶部17の内容は、クラスファイル12の実行前に、クラスファイル16の内容を解析する静的アナライザ18又は人により決定される。

本発明の一実施形態のJavaプログラム実行環境を示すソフトウェア概略ブロック図



## 【特許請求の範囲】

【請求項1】 OSプラットフォーム上で仮想マシンがその仮想マシン命令コードを解釈実行する情報処理装置において、

仮想マシン命令コード列が格納されるプログラム記憶部と、

判定情報が格納される判定情報記憶部と、

ロードされた仮想マシン命令コードをコンパイルして実行するJITコンパイラがインストールされた記憶部と、

該プログラム記憶部から仮想マシン命令コードを読み出し、該判定情報に基づいてこの仮想マシン命令コードのロード先を該仮想マシンにするか該JITコンパイラにするかを決定し、決定されたロード先に該仮想マシン命令コードをロードするという処理を繰り返し実行するプログラムローダがインストールされた記憶部と、を有することを特徴とする情報処理装置。

【請求項2】 上記判定情報は、参照命令コードを有し、上記プログラムローダが動作する前に上記判定情報記憶部に格納されており、

上記プログラムローダは、上記プログラム記憶部から読み出した仮想マシン命令コードが該参照命令コードと一致すると判定すれば、該仮想マシン命令コードを上記JITコンパイラにロードする、ことを特徴とする請求項1記載の情報処理装置。

【請求項3】 上記判定情報は、比較値とカウントとをさらに有し、

上記プログラムローダは、上記プログラム記憶部から読み出した仮想マシン命令コードが該参照命令コードと一致すると判定すれば、該カウントを変化させ、該カウントと該比較値との比較結果に基づいて該仮想マシン命令コードの上記ロード先を判定する、ことを特徴とする請求項2記載の情報処理装置。

【請求項4】 上記プログラムローダが動作する前に、上記仮想マシン命令コード列を解析して上記参照命令コードを決定する静的アナライザがインストールされた記憶部、

をさらに有することを特徴とする請求項2又は3記載の情報処理装置。

【請求項5】 仮想マシン命令コードを読み出し、判定情報に基づいてこの仮想マシン命令コードのロード先を、仮想マシン命令コードをOSプラットフォーム上で解釈実行する仮想マシンにするか、仮想マシン命令コードをOSプラットフォーム上で動的にコンパイルし実行するJITコンパイラにするかを決定し、決定されたロード先に該仮想マシン命令コードをロードする、という処理を繰り返し実行するプログラムローダが記録されていることを特徴とするコンピュータ読み取り可能な記録媒体。

## 【発明の詳細な説明】

## 【0001】

【発明の属する技術分野】本発明は、OSプラットフォーム上で、仮想マシンがその仮想マシン命令コードを解釈実行しJITコンパイラが仮想マシン命令コードを動的にコンパイルし実行する情報処理装置及びその処理を行うプログラムが記録されたコンピュータ読み取り可能な記録媒体に関する。

## 【0002】

【従来の技術】図7は、Java（登録商標）プログラム実行環境を示すソフトウェア概略ブロック図である。

【0003】プログラムとデータを含むJavaソースファイル10は、Javaコンパイラ11により、バイトコード列とデータを含むクラスファイル12に変換される。バイトコードは、JVM（Java Virtual Machine）13により解釈され実行される。JVM13は、Windows（登録商標）14上で動作しており、Windows14が持っている各種サブルーチンを利用することができる。JVM13を備えることにより、マシンに依存しないバイトコードを実行することができるという利点がある反面、オーバーヘッドが大きいという問題がある。

【0004】この問題を解決するために、Windows14上で動作するJIT（Just-In-Time）コンパイラ15を備え、クラスファイル12のバイトコードをJITコンパイラ15により実行時にコンパイル、すなわち動的にコンパイルし、その結果を記憶しておき、2回目以降このバイトコード（仮想マシンコード）を実行する場合には、再度コンパイルせずにコンパイルされたネイティブコード（ネイティブマシンコード）を実行するという方法が用いられている。

## 【0005】

【発明が解決しようとする課題】しかしながら、クラスファイル12の内容によっては、JITコンパイラ15で処理した方がJVM13で処理するよりも遅くなる場合がある。この点は、Windows14以外のOS上のJavaプラットフォーム（Javaコンパイラ11、JVM13及びJITコンパイラ15は、Windows14上で動作するJavaプラットフォームを構成している。）で動作するJavaプログラムについても同様であり、また、Smalltalkなどの他の仮想マシンプログラムについても同様である。

【0006】本発明の目的は、このような問題点に鑑み、仮想マシンのプログラムをより高速に実行することが可能な情報処理装置及びその処理を行うプログラムが記録されたコンピュータ読み取り可能な記録媒体を提供することにある。

## 【0007】

【課題を解決するための手段及びその作用効果】本発明では、OSプラットフォーム上で仮想マシンがその仮想マシン命令コードを解釈実行する情報処理装置におい

て、仮想マシン命令コード列が格納されるプログラム記憶部と、判定情報が格納される判定情報記憶部と、ロードされた仮想マシン命令コードをコンパイルして実行するJITコンパイラがインストールされた記憶部と、該プログラム記憶部から仮想マシン命令コードを読み出し、該判定情報に基づいてこの仮想マシン命令コードのロード先を該仮想マシンにするか該JITコンパイラにするかを決定し、決定されたロード先に該仮想マシン命令コードをロードするという処理を繰り返し実行するプログラムローダがインストールされた記憶部とを有する。

【0008】この情報処理装置によれば、1つ以上の仮想マシン命令コードが読み出される毎に、判定情報の内容に応じて仮想マシン又はJITコンパイラで実行されるので、仮想マシンのみ又はJITコンパイラのみで仮想マシン命令コードを実行する場合よりも処理時間が短縮される。

【0009】本発明の他の目的、構成及び効果は以下の説明から明らかになる。

【0010】

【発明の実施の形態】以下、図面を参照して本発明の一実施形態を説明する。

【0011】図1は、本発明の一実施形態のJavaプログラム実行環境を示すソフトウェア概略ブロック図である。

【0012】このソフトウェアは、コンピュータや携帯電話などの情報処理装置にインストールされている。

【0013】本実施形態では、クラスファイル12のバイトコードを、その内容に応じてJVM13で解釈実行し又はJITコンパイラ15で動的にコンパイルし実行するという処理を繰り返し行うことにより、JVM13のみ又はJITコンパイラ15のみで実行する場合よりも高速化を図っている。

【0014】バイトコードのロード先をJVM13とするかJITコンパイラ15とするかは、プログラムローダ16により、判定情報記憶部17の内容に基づいて判定される。判定情報記憶部17の内容は、クラスファイル12の実行前に、クラスファイル12の内容を解析する静的アナライザ18又は人（プログラム開発者やシステム運用者など）により決定される。

【0015】図2は、図1のクラスファイル12の実行手順を示す概略フローチャートである。

【0016】（S1）プログラムローダ16は、クラスファイル12から次のバイトコードを読み出す。

【0017】（S2）プログラムローダ16は、このコードが終了コードであれば処理を終了し、そうでなければ次のステップS3へ進む。

【0018】（S3）このバイトコードがサブルーチンコールでなければステップS4へ進む。このバイトコードがサブルーチンコールであれば、プログラムローダ1

6は、クラスファイル12からこれに関連したバイトコード群（現在のバイトコードで呼び出されるサブルーチン）をクラスファイル12から読み出し、バイトコード群が、過去にJITコンパイラ15でコンパイル済みでありその結果がメモリに格納されていればステップS7へ進む、そうでなければステップS4へ進む。

【0019】（S4）ステップS1で読み出したバイトコードがサブルーチンコールでなければステップS5へ進む。このバイトコードがサブルーチンコールであれば、プログラムローダ16は、判定情報記憶部17の内容を参照し、バイトコード群をコンパイルするかどうかを後述のようにして判定する。コンパイルしないと判定した場合にはステップS5へ進む、すると判定した場合にはステップS6へ進む。

【0020】（S5）プログラムローダ16は、バイトコードをJVM13にロードする。JVM13は、このバイトコードを解釈し、得られたネイティブコードをMPUに供給して実行させる。次にステップS1へ戻る。

【0021】（S6）プログラムローダ16は、バイトコード群をJITコンパイラ15にロードする。JITコンパイラ15は、このバイトコード群をコンパイル（ネイティブコードに変換）する。

【0022】（S7）JITコンパイラ15は、変換されているネイティブコードをMPUに供給して実行させる。次に、ステップS1へ戻る。

【0023】図1において、JVM13及びJITコンパイラ15はいずれもWindows14上で動作し、Windows14が持っているサブルーチンを呼び出すことによりこれを利用することができる。

【0024】図3は、クラスファイル12のバイトコードを実行中にWindows14のグラフィック描画ライブラリが呼び出され、その実行中にさらにデバイスドライバが読み出される場合を示している。このような処理では、グラフィック描画ライブラリ及びデバイスドライバでの処理時間が比較的に長いので、グラフィック描画ライブラリを呼び出すバイトコード群をJVM13で実行してもJITコンパイラ15で実行してもほぼ同じになる。この場合、JITコンパイラ15で実行すると、コンパイルされたネイティブコード列を記憶しておく必要があるため、プログラムローダ16はこのバイトコード群をJVM13にロードする。

【0025】図4の（A）及び（B）はいずれも、あるバイトコード群がJVM13で繰り返し実行される場合とJITコンパイラ15で繰り返し実行される場合の処理時間の比較を示す。

【0026】JVM13で実行する場合には、毎回同じ処理時間 $t_1$ となるが、JITコンパイラ15で実行する場合にはバイトコード群が1回コンパイルされると2回目以降はコンパイルが省略されるので、2回目以降の処理時間 $t_3$ は1回目の処理時間（ $t_2 + t_3$ ）よりも

コンパイル時間 $t_2$ だけ短くなる。なお、時間 $t_3$ にもオーバーヘッドが含まれている。

【0027】図4(A)では、あるバイトコード群を1回のみ実行する場合には、JVM13を用いた方が高速であるが、このバイトコード群を2回実行する場合にはJVM13とJITコンパイラ15のいずれを用いてもほぼ同じとなり、このバイトコード群を3回以上実行する場合にはJITコンパイラ15を用いた方が処理時間を短縮することができることを示している。また、図4(B)は、あるバイトコード群を1回のみ実行する場合にはJVM13とJITコンパイラ15のいずれを用いてもほぼ同じ時間となるが、このバイトコード群を2回以上実行する場合には、JITコンパイラ15を用いた方が処理時間を短縮できることを示している。

【0028】このような解析が、図1の静的アナライザ18又は人により行われて、判定情報記憶部17に格納すべき情報が決定される。

【0029】図5は、判定情報記憶部17の内容の具体例と、これに関係した部分とを示す説明図である。

【0030】判定情報記憶部17には、JITコンパイラ15で実行される可能性のあるサブルーチンコールバイトコードBC $\alpha$ ~BC $\delta$ (引数を設定するバイトコードは含まれない。)が参照バイトコードとして予め格納されている。各参照バイトコードには、後述のフラグF、比較値N及びカウントkの情報が付加されている。フラグF及び比較値Nは、参照バイトコードと同様に、静的アナライザ18又は人により決定される。

【0031】図6は、クラスファイル12から読み出された1つのサブルーチンコールバイトコードBC $i$ に対する図5のプログラムローダ16の処理を示すフローチャートである。

【0032】(S9)判定情報記憶部17内の各参照バイトコードに対応したカウントkをゼロクリアする。

【0033】(S10)バイトコードBC $i$ を判定情報記憶部17の次の参照コードと比較する。最初は、バイトコードBC $i$ を参照コードBC $\alpha$ と比較する。

【0034】(S11)判定情報記憶部17に次の参照バイトコードがなければステップS12へ進み、そうでなければステップS13へ進む。

【0035】(S12)バイトコードBC $i$ をJVM13にロードし、処理を終了する。

【0036】(S13)比較結果が一致すればステップS14へ進み、そうでなければステップS10へ戻る。

【0037】(S14)F='0'であればステップS15へ進み、F='1'であればステップS18へ進む。

【0038】(S15)カウントkを1だけインクリメントする。

【0039】(S16)N<N<sub>max</sub>であればステップS10へ戻り、N=N<sub>max</sub>であれば次のステップS17へ進む。

【0040】(S17)フラグFを'1'にする。

【0041】(S18)バイトコードBC $i$ に関連したバイトコードをクラスファイル12から読み出し、これとバイトコードBC $i$ とからなるバイトコード群をJITコンパイラ15にロードする。

【0042】このような処理により、バイトコードBC $i$ とF='1'の参照バイトコードとが一致する場合にはバイトコードBC $i$ を含むバイトコード群がJITコンパイラ15で実行され、バイトコードBC $i$ とF='0'の参照バイトコードとが一致する場合にはこのバイトコードBC $i$ がN回以上出現したときにバイトコードBC $i$ を含むバイトコード群がJITコンパイラ15で実行される。例えば、図4(A)の場合にはF='1'、N=2であり、バイトコードBC $i$ が2回以上出現したときに、その後3回以上出現すると予測して、バイトコードBC $i$ を含むバイトコード群をJITコンパイラ15で実行する。図4(B)の場合にはF='0'であり、バイトコードBC $i$ が出現すればこれを含むバイトコード群を必ずJITコンパイラ15で実行する。

【0043】本実施形態によれば、バイトコードが上述のようにその内容に応じてJVM13又はJITコンパイラ15で実行されるので、JVM13のみ又はJITコンパイラ15のみでバイトコードを実行する場合よりも処理時間が短縮される。

【0044】なお、本発明には外にも種々の変形例が含まれる。

【0045】例えば、バイトコード群をJITコンパイラ15で実行した場合の1回目の時間と2回目の時間 $t_2$ 及び $t_3$ を予め判定情報記憶部17に格納しておき、クラスファイル12の実行中において、このバイトコード群が最初に出現したときプログラムローダ16はこれをJVM13にロードし、その実行時間 $t_1$ を測定してこれを判定情報記憶部17に格納し、 $t_1 \sim t_3$ の値に基づいて比較値Nを決定するようにしてもよい。

【0046】図5において、フラグFを用いずに、比較値Nとカウントkとを用い、kの初期値を0とし、F='1'に対応したNの値を1とし、バイトコードBC $i$ と参照バイトコードとが一致した場合にkを1だけインクリメントし、k=Nのときバイトコード群をJITコンパイラ15にロードするようにしてもよい。また、kの初期値をNとし、F='1'に対応したNの値を1とし、バイトコードBC $i$ と参照バイトコードとが一致した場合にkを1だけデクリメントし、k=0のときバイトコード群をJITコンパイラ15にロードするようにしてもよい。

【0047】本発明はJava言語以外の仮想マシンにも適用される。仮想マシンは、Windows 14以外のOS上で動作するものであってもよい。

【図面の簡単な説明】

(5)

【図1】本発明の一実施形態のJavaプログラム実行環境を示すソフトウェア概略ブロック図である。

【図2】図1のクラスファイルの実行手順を示す概略フローチャートである。

【図3】クラスファイルのバイトコードを実行中にWindowsのグラフィック描画ライブラリが呼び出され、その実行中にさらにデバイスドライバが読み出される場合を示す説明図である。

【図4】(A)及び(B)はいずれも、あるバイトコード群がJVMで繰り返し実行される場合とJITコンパイラで繰り返し実行される場合の処理時間の比較を示す線図である。

【図5】図1の判定情報記憶部の内容の具体例と、これに関係した部分とを示す説明図である。

【図6】クラスファイルから読み出された1つのサブルーチンコールバイトコードBCiに対する図5のプログラムローダの処理を示すフローチャートである。

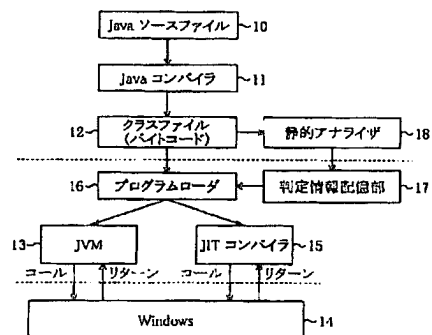
【図7】従来のJavaプログラム実行環境を示すソフトウェア概略ブロック図である。

【符号の説明】

10 Javaソースプログラム  
11 Javaコンパイラ  
12 クラスファイル  
13 JVM  
14 Windows  
15 JITコンパイラ  
16 プログラムローダ  
17 判定情報  
18 静的アナライザ  
F フラグ  
N 比較値  
k カウント  
BCi バイトコード

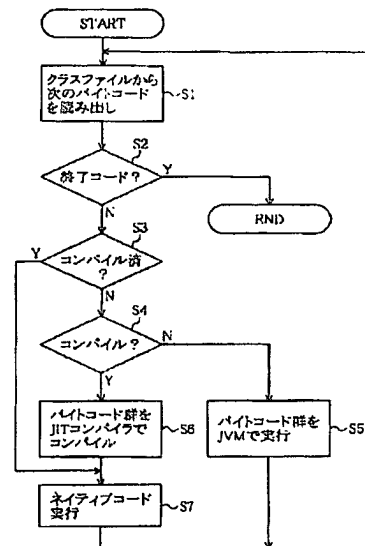
【図1】

本発明の一実施形態のJavaプログラム実行環境を示すソフトウェア概略ブロック図



【図2】

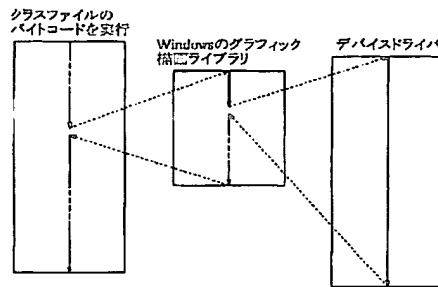
図1のクラスファイルの実行手順を示す概略フローチャート



(6)

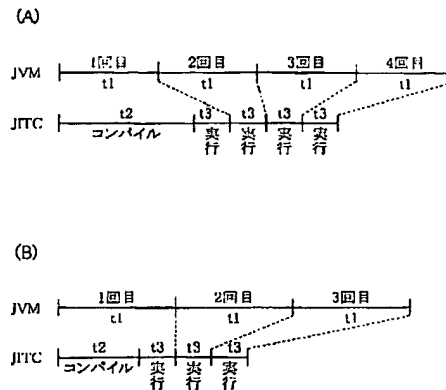
【図3】

クラスファイルのバイトコードを実行中にWindowsのグラフィック描画ライブラリが呼び出され、その実行中にさらにデバイスドライバが読み出される場合を示す説明図



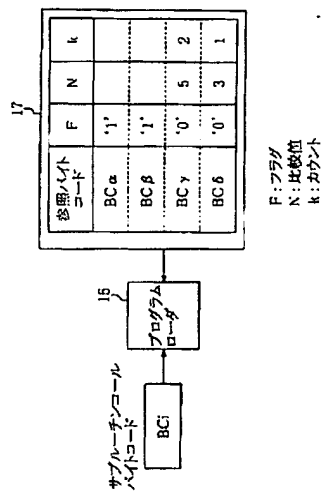
【図4】

(A) 及び (B) はいずれも、あるバイトコード群が JVM で繰り返し実行される場合と JIT コンパイラで繰り返し実行される場合の処理時間の比較を示す線図



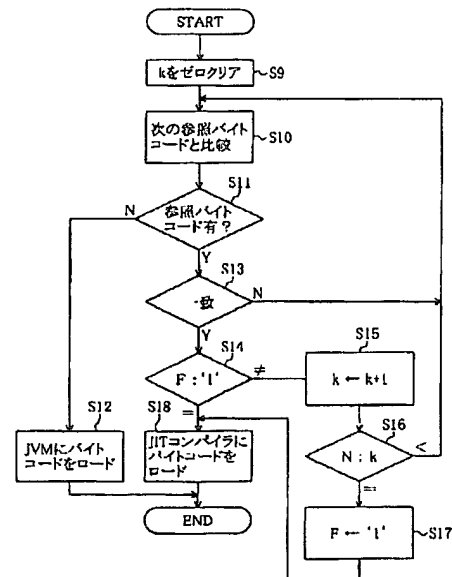
【図5】

図1の判定情報記憶部の内容の具体例と、これに関連した部分とを示す説明図



【図6】

クラスファイルから読み出された1つのサブルーチンコールバイトコードBCiに対する図5のプログラムローダの処理を示すフローチャート



【図7】

従来のJavaプログラム実行環境を示すソフトウェア  
模塊ブロック図

